TD/TP 6 : Graphe

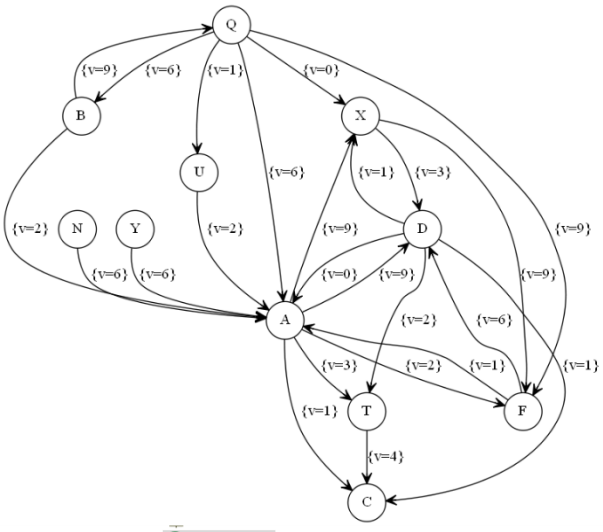
Objectif du TP

L'objectif de cette séance est de manipuler des structures de données récursives (majoritairement des *graphes*) et des algorithmes les manipulant.

EXERCICES SUR PAPIER !

Vous ***devez*** répondre aux exercices suivants **sur papier** et y travailler **seul**. Cela évite le bavardage, donc le bruit et favorise grandement la ***concentration*** des autres. Une fois que vous avez **fini un exercice**, vous vous manifestez auprès de votre enseignant pour qu'il ***juge*** votre travail sur une échelle de 0 à **4 points** (0=aucun travail, …, 4=exercice complètement juste sans assistance de l'enseignant).

Exercice 1 Plus court chemin

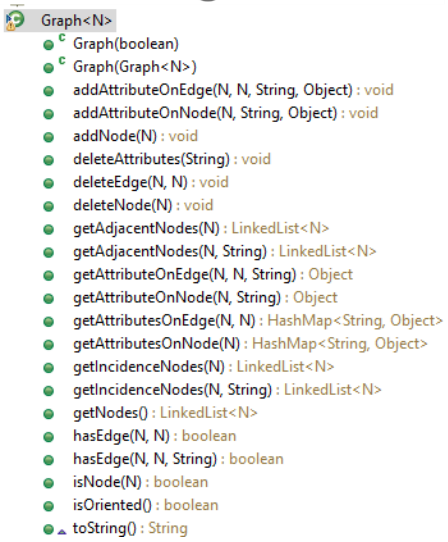


En considérant le graphe ci-contre, quel est le plus court chemin entre :

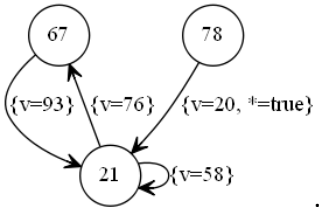
a) C et X ?
b) Q et F ?
c) Faites une trace de l'algorithme A* avec ce graphe en utilisant une heuristique nulle.

Exercice 2 Utilisation de la classe Graph

Récupérez le fichier `Graph.java` qui contient le code correspondant au type `Graphe` dont les nœuds sont de type `N` représenté ci-contre. La classe contient sa propre documentation.



a) Donner le code permettant de créer le graphe suivant



.

b) Ecrire la méthode `createRandomDirectedGraph` qui crée un graphe orienté aléatoire contenant exactement `MAXNumberOfNode` nœuds et au maximum `MAXNumberOfEdge` arêtes. Les nœuds seront des entiers et les valeurs des arêtes seront réelles et associées à la clé « v ».

c) On souhaite identifier les composantes connexes d'un graphe. Pour les trouver, on lance une exploration à partir d'un sommet `s` quelconque du graphe, ce qui permet de trouver la composante connexe de `s`. Tous les sommets de cette composante sont marqués, et tous les autres sommets sont non marqués. On relance donc une exploration à partir d'un des sommets non encore marqué pour trouver sa composante, et on itère de la sorte tant que tous les sommets du graphe ne sont pas marqués. Le nombre d'itérations correspond exactement au nombre de composantes connexes du graphe.

EXERCICES SUR MACHINES !

À partir de maintenant, vous **pouvez** (je n'ai **pas** dit *devez* !) implémenter l'exercice sur machine. Il est fortement conseillé de résoudre le problème sur papier avant de l'implémenter dans le langage de votre choix (C, D, Java, Python, Julia, … https://fr.wikipedia.org/wiki/Liste_de_langages_de_programmation).

Exercice 3 Plus court chemin dans un graphe

Dans une nouvelle classe constante `GraphAlgorithms`, créez une méthode `LinkedList<N> shortestPath(Graph<N> g, N start, N goal, String key4dist, Heuristic<N> h_score)` qui calcule le plus court chemin dans le graphe en utilisant l'algorithme vu en cours.

POUR ALLER PLUS LOIN

Sources pour ce TP